95-411 [WGM 1893] Patent

UNITED STATES UTILITY PATENT APPLICATION

FOR

ARRANGEMENT FOR CONTROLLING AND LOGGING VOICE ENABLED WEB APPLICATIONS USING EXTENSIBLE MARKUP LANGUAGE DOCUMENTS

INVENTORS:

Ryan Alan Danner of Richmond, Virginia
and
Steven J. Martin of Richmond, Virginia

PREPARED BY:

Leon R. Turkevich, Esq.
2000 M STREET, N.W., 7th Floor
WASHINGTON, D.C. 20036-3307
(202) 261-1059

ARRANGEMENT FOR CONTROLLING AND LOGGING VOICE ENABLED WEB APPLICATIONS USING EXTENSIBLE MARKUP LANGUAGE DOCUMENTS

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from provisional application No. 60/152,316, filed September 3, 1999, the disclosure of which is incorporated in its entirety herein by reference.

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to generating and executing voice enabled web applications within a hypertext markup language (HTML) and hypertext transport protocol (HTTP) framework.

DESCRIPTION OF THE RELATED ART

The evolution of the public switched telephone network has resulted in a variety of voice applications and services that can be provided to individual subscribers and business subscribers. Such services include voice messaging systems that enable landline or wireless subscribers to record, playback, and forward voice mail messages. However, the ability to provide enhanced services to subscribers of the public switched telephone network is directly affected by the limitations of the public switched telephone network. In particular, the public switched telephone network operates according to a protocol that is specifically designed for the transport of voice signals; hence any modifications necessary to provide enhanced services can only be done by switch vendors that have sufficient know-how of the existing public switched telephone network infrastructure.

An open standards-based Internet protocol (IP) network, such as the World Wide Web, the Internet, or a corporate intranet, provides client-server type application services for clients by enabling the clients to request application services from remote servers using standardized protocols, for example hypertext transport protocol (HTTP). The web server application environment can include web server software, such as Apache, implemented on a computer system attached to the IP network. Web-based applications are composed of HTML pages, logic, and database functions. In addition, the web server may provide logging and monitoring capabilities.

In contrast to the public switched telephone network, the open standards-based IP network has enabled the proliferation of web based applications written by web application developers using ever increasing web development tools. Hence, the ever increasing popularity of web applications and web development tools provides substantial resources for application developers to develop robust web applications in a relatively short time and an economical manner. However, one important distinction between telephony-based applications and web-based applications is that telephony-based applications are state aware, whereas web-based applications are stateless.

In particular, telephony applications are state aware to ensure that prescribed operations between the telephony application servers and the user telephony devices occur in a prescribed sequence. For example, operations such as call processing operations, voicemail operations, call forwarding, etc., require that specific actions occur in a specific sequence to enable the multiple components of the public switched telephone network to complete the prescribed operations.

The web-based applications running in the IP network, however, are state-less and transient in nature, and do not maintain application state because application state requires an interactive communication between the browser and back-end database servers accessed by the browsers via a HTTP-based web server. However, an HTTP server provides asynchronous execution of HTML applications, where the web applications in response to reception of a specific request in the form of a URL from a client, instantiate a program configured for execution of the specific request, send an HTML web page back to the client, and terminate the program instance that executed the specific request. Storage of application state information in the form of a "cookie" is not practical because some users prefer not to enable cookies on their browser, and because the passing of a large amount of state information as would normally be required for voice-type applications between the browser and the web application would substantially reduce the bandwidth available for the client.

SubA1

Commonly-assigned, copending application serial number 09/480,485, filed January 11, 2000, entitled Application Server Configured for Dynamically Generating Web Pages for Voice Enabled Web Applications (Attorney Docket 95-309), the disclosure of which is incorporated in its entirety herein by reference, discloses an application server that executes a voice-enabled web application by runtime execution of extensible markup language (XML) documents that define the

Sub A1

~~voice-enabled web application to be executed. The application server includes a runtime environment that establishes an efficient, high-speed connection to a web server. The application server, in response to receiving a user request from a user, accesses a selected XML page that defines at least a part of the voice application to be executed for the user. The XML page may describe any one of a user interface such as dynamic generation of a menu of options or a prompt for a password, an application logic operation, or a function capability such as generating a function call to an external resource. The application server then parses the XML page, and executes the operation described by the XML page, for example dynamically generating an HTML page having voice application control content, or fetching another XML page to continue application processing. In addition, the application server may access an XML page that stores application state information, enabling the application server to be state-aware relative to the user interaction. Hence, the XML page, which can be written using a conventional editor or word processor, defines the application to be executed by the application server within the runtime environment, enabling voice enabled web applications to be generated and executed without the necessity of programming language environments.~~

Hence, web programmers can write voice-enabled web applications, using the teachings of the above-incorporated application serial number 09/480,485, by writing XML pages that specify respective voice application operations to be performed. The XML documents have a distinct feature of having tags that allow a web browser (or other software) to identify information as being a specific kind or type of information.

One particular concern in the execution of voice-enabled web applications involves the logging of events that occur, during execution of the applications, into a log file and the ability to analyze the log file. Specifically, deployment of voice-enabled web applications in a commercial environment may require that the log file can be analyzed by various telephony management systems such as billing, monitoring, scripting, etc.. However, existing methods of writing structured log files typically involves performing a "data dump" during an application operation with little or no structure to the data, rendering it difficult or impossible to effectively utilize the log data for voice enabled web applications. For example, conventional web servers have log facilities that are based on the logging of page or CGI information. The logging of the page or CGI information, however, is not sufficient for the management of voice enabled applications, since telephony management systems require more complex interaction, tracing, session reporting, and billing log information than generated by conventional web servers. Hence, the logging facilities of conventional web servers may prove inadequate to support the commercial deployment of voice-enabled web applications.

Another concern in the execution of voice-enabled web applications involves developing voice-enabled web applications using XML documents in a manner that provides sufficient flexibility for modification of the voice-enabled web applications with minimal changes to the XML documents defining the applications. In particular, care must be taken that the specificity of the XML documents defining the executable application do not limit the ability to easily modify the executable application.

SUMMARY OF THE INVENTION

There is a need for an arrangement that enables voice applications to be defined using extensible markup language (XML) documents in a manner that provides flexible control of the voice applications. In particular, there is a need for an arrangement that enables the modification of existing application parameters of a voice-enabled web application, or the addition of new application parameters, without affecting unmodified application parameters.

There is also a need for an arrangement that enables the generation of structured log files for voice-enabled web applications, where the structured log files can be analyzed using the structured nature of XML documents.

These and other needs are attained by the present invention, where control data for a voice-enabled web application, and log files that record events that occur during execution of the voice-enabled web application, are generated and processed using an XML tag format. The voice-enabled web application is defined, for example, using a first set of extensible markup language (XML) documents that define the voice application operations to be performed within the voice application. A second set of XML documents specify application parameters and control information to be used by the application runtime environment for execution of the first set of XML documents. The second set of XML documents enable the application parameters and control information to be stored separately from the first set of XML documents, eliminating the necessity for overly-specific XML documents to define a voice-enabled web application. In addition, the second set of XML documents enables the application server to maintain a generic application runtime environment, enabling applications to share common control information and provide personalized services for subscribers based on respective user specific control attributes.

The generation of log files using an XML tag format enables the log files to use a standardized XML structure that includes log element type, log element attribute, and log element data information. Hence, logs may be written for individual user sessions and overall application information, where the XML log tags may be of sufficient descriptive nature as to be understood using any XML viewer. Alternatively, the logs may be analyzed by custom log parser configured for locating prescribed XML tags related to a corresponding operation, for example billing, trace routing, etc.. Hence, logging information can be more easily analyzed based on parsing the XML tags within the XML log documents.

One aspect of the present invention provides a method in an executable system for controlling execution of an executable voice application. The method includes storing an extensible markup language (XML) control document specifying at least one application control parameter for execution of the executable voice application in an application runtime environment generated by the executable system, and parsing the XML control document for execution of the executable voice application by the application runtime environment according to the at least one application control parameter.

Another aspect of the present invention provides a method in an executable system for generating a log file that specifies an occurrence of an event in response to execution of an executable voice application. The method includes generating an XML log document having a log entry that specifies the occurrence of the event in response to the execution of the executable voice application, the generating step including generating first, second and third XML tags specifying a log element type, a log element attribute, and a log element data for the event, respectively, and outputting the XML log document for storage on a tangible medium.

Additional advantages and novel features of the invention will be set forth in part in the description which follows and in part will become apparent to those skilled in the art upon examination of the following or may be learned by practice of the invention. The advantages of the present invention may be realized and attained by means of instrumentalities and combinations particularly pointed out in the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Reference is made to the attached drawings, wherein elements having the same reference numeral designations represent like elements throughout and wherein:

Figure 1 is a block diagram illustrating in a system configured for the generation and usage of XML control documents and XML log documents during execution of voice applications according to an embodiment of the present invention.

Figure 2 is a diagram illustrating in further detail the application runtime environment of Figure 1.

Figure 3 is a flow diagram illustrating the method of controlling execution of an executable voice application and generating a log file according to an embodiment of the present invention.

Figures 4A and 4B are diagrams illustrating an XML control document and an XML log document, respectively.

BEST MODE FOR CARRYING OUT THE INVENTION

The ability to provide unified voice messaging services via an IP network enables existing web servers on the World Wide Web or in corporate intranets to support telephone applications on a scalable and economic platform. Moreover, the use of XML documents in defining executable voice applications enables use of open standards that permits web programmers to use forms-based web programming techniques to design and implement voice telephone applications, without the necessity of programming using conventional programming languages. In addition, the use of XML documents in defining executable voice applications provides a structured application tool that enables a user to easily customize a voice application by making relatively minor changes to the XML documents. Additional details regarding the use of XML documents to define voice-enabled web applications is described in commonly-assigned, copending application serial number 09/501,516, filed February 9, 2000, entitled Arrangement for Defining and Processing Voice-Enabled Web Applications Using Extensible Markup Language Documents (attorney docket 95-410), the disclosure of which is incorporated in its entirety herein by reference.

The disclosed embodiment is directed to an arrangement for controlling execution of an executable voice application, and generating a log file that specifies an occurrence of an event in response to execution of the executable voice application, using extensible markup language (XML) documents. The structured nature of XML documents enables an application developer to provide structured flexibility to voice-enabled web applications, enabling the voice-enabled web applications to be easily modified even during execution in an application runtime environment.

The disclosed embodiment further enhances the ability to provide a structured model for the commercial deployment of voice-enabled web applications by providing a method in which web application control data and log output can be created and processed using an XML tag format. In particular, the disclosed embodiment provides extremely flexible control of a voice-enabled web application by separating application control parameters, stored in XML control documents, from application operations such as user interface operations, logic operations, and/or function operations. The separation of application control parameters from application operations by storing the application control parameters in separate XML documents enables new application control parameters to be added without affecting existing versions of the voice-enabled web application. In addition, existing application control parameters can be deleted without affecting future versions of the voice-enabled web application.

Moreover, the use of XML control documents to specify application control parameters facilitates the sharing of common control information between multiple voice-enabled web applications, since each voice-enabled web application does not need to rely on the control parameters to be specified in a specific state-dependent sequence. In addition, application control parameters can be described using the XML tag format, enabling user-specific control attributes to be specified within the XML control documents. Hence, a given XML control document may either be shared among multiple voice-enabled web applications for basic control parameters, or may be generated for a specific subscriber to store subscriber-specific application control parameters.

The disclosed embodiment also provides an arrangement for generating XML log documents, wherein each XML log document has at least one log entry that specifies an occurrence of an event in response to execution of the executable voice application. The storage of a log entry in XML format enables the use of the XML structure to define XML tags that specify log element types, log element attributes, and log element data information. The logs may be written for both individual sessions and overall application information. Moreover, the XML log tags are sufficiently descriptive that any XML viewer can be used to understand the log information. Alternatively, custom log parsers can be easily written, using standard XML editor tools, to gather and format log information relevant to a given operation, for example operations monitored by telephony management systems such as billing, monitoring, scripting, call processing management, or traffic management.

Figure 1 is a block diagram illustrating a system configured for executing applications defined by XML documents based on application control parameters specified in XML control documents, and generating XML log documents for analysis according to an embodiment of the present invention. The system 10 includes an application server 12, a Web server 14, an XML document database 16, and an XML document registry 18. The XML document database 16 is configured for storing the XML documents that define the applications, and the XML control documents that specify application control parameters for execution of the executable voice applications. The XML document registry 18 is configured for storing XML documents that specify application state information for respective user sessions. The web server 14, for example an Apache server, receives HTTP requests from a client 5 via the Internet and forwards to the client HTML-based web pages dynamically generated by the application server 12. In particular, the application server 12 is configured for executing applications defined by stored XML documents, also referred to generally as web applications, in response to the HTML requests from the client. The application server 12 is implemented as a server executing a PHP hypertext processor with XML parsing and processing capabilities, available open source at http://www.php.net.

Four types of XML documents are used by the application server 12 to execute web applications: menu documents, activity documents, decision documents, and "brownies". The menu documents, activity documents, and decision documents are XML documents, stored in the document database 16, that define user interface and boolean-type application logic for a web application, hence are considered "executable" by the application server 12. The brownie document, stored in the registry 18, is an XML data record used to specify application state and user attribute information for a given XML application during a user session.

Hence, the XML documents define user interface logistics and tie services and application server events together in a meaningful way, forming a coherent application or sets of applications.

The document database 16 also is configured for storing XML control documents that specify application control parameters for execution of the voice-enabled Web applications by the application server 12. The document database 16 may also be used for storage of XML log documents that specify the occurrence of events in response to execution of the voice-enabled web applications by the application server 12. The XML control documents, as well as the application-defining documents, can be created and edited by an XML document editor 5, for example a web browser having editing capabilities. In particular, the browser 5 may be provided a form-based menu generated by the application server 12; the user of the browser 5 completes the form, and posts the form back to a prescribed URL at the web server 14; the application server 12, in response to receiving the form having the prescribed URL, retrieves the information input to the posted form, and stores the retrieved information into an XML control document for storage in the document database 16.

The application server 12 includes an XML parser 20 configured for parsing the XML documents stored in the XML document database 16, for example the application-defining XML documents, the XML control documents, or the XML log documents. The XML parser 20 also is configured for parsing the XML documents (i.e., "brownies") stored in the registry 18 and configured for specifying the state and attributes for respective user sessions. The application server 12 also includes a high speed interface 22 that establishes a high-speed connection between the application server 12 and the web server 14. For example, the PHP hypertext processor includes a high-speed interface for Apache Web servers.

The application server 12 also includes a runtime environment 24 for execution of the parsed XML documents. Figure 2 is a diagram illustrating in detail the application runtime environment 24 of Figure 1, emphasizing the use of XML control documents to define application context. The application runtime environment 24 executes the voice-enabled web applications 60a and 60b by first parsing an XML control document 62 that specifies the necessary application control parameters for execution of the voice-enabled web applications 60a and 60b. Hence, the XML control document 62 provides basic runtime control defaults for the application runtime environment 24. In addition, the application runtime environment 24 may parse a second XML control file 64 that provides user-specific attributes overlying the basic control defaults specified in the XML control document 62. Hence, the application runtime environment 24 may parse a first XML control document 62 for basic runtime control defaults for the application runtime environment 24, followed by parsing of a second XML control document 64 for user-specific attributes for a corresponding subscriber 66. Hence, the application runtime environment 24 obtains basic runtime control defaults and user-specific attributes from stored XML control documents, enabling the stored XML control documents to provide context information to the application runtime environment 24. The context information is then used by the application runtime apartment 24 for execution of the application-defining XML documents.

The runtime environment 24 parses the application-defining XML document for execution of the web application. As shown in Figure 1, the runtime environment 24 may selectively execute any one of a user interface operation 26, a logic operation 28, or a procedure call 30 as specified by the parsed application-defining XML document by executing a corresponding set of executable functions based on the rule set for the corresponding operation and based on the context information obtained by the application runtime environment 24. In particular, the application runtime environment 24 includes a tag implementation module 32 that implements the XML tags parsed by the XML parser 20. The tag implementation module 32 performs relatively low-level operations based on the obtained context information, for example dynamically generating an XML menu page using executable functions specified by a menu rule set in response to detecting a menu tag, performing a logical operation using executable functions specified by a logic rule set in response to a decision tag, or fetching an audio (.wav) file in response to detecting a sound tag. Hence, the tag implementation module 32 implements the tag operations that are specified within the XML framework of the stored XML documents. As described above, execution of the application operations by the runtime environment 24 is based on the control information from the parsed XML control documents, including basic runtime control defaults and/or user-specific attributes.

The application server also includes a log module 50 configured for generating an XML log document 70 for storage in the database 16. As described below, the log module 50 generates a log document for an event that occurred during execution of the voice-enabled web application in the application runtime environment, and generates XML tags that specify the type of event, the attribute of the event, and data for the event.

The application server 12 also includes a set of libraries 34 that may be implemented as dynamically linked libraries (DLLs) or application programming interface (API) libraries. The libraries 34 enable the runtime environment 24 to implement the procedures 30 as specified by the appropriate XML document. For example, the application server 12 may issue a function call to one of a plurality of IP protocol compliant remote resources 40, 42, or 44 according to IMAP protocol, LDAP Protocol, or SMTP protocol, respectively. For example, the PHP hypertext processor includes executable routines capable of accessing the IMAP or LDAP services. As described in further detail below, the application runtime environment 24 accesses the libraries 34 based on the context information obtained from the XML control documents.

Figure 3 is a diagram illustrating a method of generating XML control documents and XML logging documents according to an embodiment of the present invention. The disclosed method can be implemented by a processor which executes instructions stored on a computer readable medium. Figure 3 also illustrates the usage of XML control documents for execution of voice-enabled web applications by the application runtime environment 24. The method begins in step 80, where the application runtime environment 24 executes a master configuration XML document in the document database 16, that specifies all the necessary XML control documents needed for execution of voice-enabled web applications. As readily apparent from the foregoing, the master configuration XML document is generated by a user of the browser 5, using an XML document editor tool, to specify all necessary XML control documents 62. The application runtime environment 24 then loads each XML control document 62 specified in the master configuration XML document in step 82 and parses the loaded XML control document to obtain the control parameters.

Figure 4A is a diagram illustrating an exemplary XML control document 62 having application control parameters for execution of a voice-enabled web application. The XML control document 62 includes XML tags that specify certain attributes of an application control parameter. For example, the XML control document 62 includes services tags 92a, 92b and 92c that specify service location information (e.g., IP address, host name, and port) necessary for the application runtime environment 24 to access IMAP, LDAP, and text-to-speech services, respectively.

The application runtime environment 24, upon loading the XML control documents in step 82, obtains the necessary context information for execution of the application-defining XML documents in response to user requests. If desired, the application runtime environment 24 may also be configured for accessing user-specific attributes in step 84: in this case, the application runtime environment 24 accesses the corresponding user-specific XML control document 64 in response to receiving a request from the subscriber to the voice-enabled web application to obtain the user-specific context information. Hence, the XML control documents 62 can be considered a basic level of control information necessary for execution of a voice-enabled web application; the user-specific XML control documents 64 can be considered the next higher-level of control information necessary for execution of a voice-enabled web application for a corresponding subscriber, and the application-defining XML documents can be considered the highest level of abstraction with respect to execution of the voice-enabled web applications by the application runtime environment 24.

Use of the XML control documents 62 and 64 provides additional flexibility for an application developer to develop voice-enabled web applications. For example, the application developer may use specific elements for searching the LDAP database 42 (e.g., "billing number" or "subscriber ID"), however the application developer may wish to remove the attribute names from the actual database so that the routine can still be implemented using another database system. Use of an XML control file enables the application developer to specify the internal name for the search attribute, and the implemented name for the alternative database system. Hence, use of the XML control documents 62 and/or 64 provides substantial flexibility enabling application developers to easily modify application behavior as necessary.

During execution of the application-defining XML documents based on the XML control documents 62 and 64, the application runtime environment 22 calls a logging function in the log module 50 in step 86 in response to parsing a prescribed XML tag, within one of the application-defining XML documents, that specifies a logging operation. The log module 50 generates the XML log document 70 in step 88 based on the log entry parameters that may be specified in the application-defining XML documents, and stores the log document 70 in the document database 16 in step 90.

Figure 4B is a diagram illustrating a sample log file 70 generated by the log module 50 in step 88. The log file 70 includes a log file tag 72 that specifies a name attribute 74 and an application attribute 76 that specifies the application having generated the log file 70 during execution by the application runtime environment 24. The log file 72 also includes log entry tags 78a and 78b that specify a standard log entry 80a and an error log entry 80b, respectively. Each log entry 80 includes a relevance tag 82 that specifies a log element type (e.g., BILLING, TRACE, NOTIFY) for the corresponding log entry 80: in other words, the relevance tag 82a specifies the log element types BILLING and TRACE, indicating that the log entry 80a is relevant for billing information and trace information. Hence, the log file 70 can be parsed by the XML parser 20 during execution of an analysis routine configured for locating log entries related to billing information, trace information, or the like. Consequently, the analysis routine can quickly identify XML log documents 70 that are relevant for the analysis under consideration.

The sample log file 70 also includes a log code XML tag 84 that specifies a prescribed log code based on a log code index. In particular, the log code may be referenced to a log code index that is used by the application runtime environment 24 to correlate the log code to the corresponding event. The sample log file 70 also includes additional tags 86 (e.g., PARAMETERS, TEXT) that specify the log element data. Hence, the structure nature of the log file 70 in XML format enables the log data to be more highly structured, such that the log file 70 can be parsed during execution of an analysis application configured for reviewing log files for selected log data; the analysis application can determine whether data is relevant based on the corresponding relevance tags 82, such that nonrelevant data can be ignored based on the corresponding XML tag. In addition, logs may be written for both individual subscriber sessions and overall application information. The logging tags are descriptive enough to understand the log information using any XML viewer, although a custom log parser can be written using standard XML parsing functions to gather and format the log information.

A particular feature of using XML documents for execution of voice-enabled web applications is that the application runtime environment 24 does not maintain a persistent state, but rather executes a prescribed application operation, sends media information back to the subscriber in the form of an HTML-based web page via the Web server 14, and then returns to a restful state. Hence, a user of the editor 5 can concurrently edit application control parameters for an XML control document 62 or 64 while conducting a subscriber session with the application server 12. In particular, the application server 12 merely performs an operation in the application runtime environment to respond to a subscriber request and then returns to a restful state, although the browser 5 provides the user the appearance that a voice-enabled web application and the editing of XML control documents can occur simultaneously. Hence, the use of XML for defining the control documents and the application-defining XML documents is particularly effective in providing the appearance of real time execution, enabling a programmer to instantaneously check an input value for debugging purposes.

While this invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not limited to the disclosed embodiments, but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.